Messaging system for delivering data in the form of portable message formats between message clients.

Field of the invention:

The invention relates to techniques for the delivery of electronic messages between
5   hardware or software components over any kind of network (wired and wireless),
between any kind of devices.

Background of the invention:

Message oriented middleware (MOM) has been available for many years. In October
1998, an industry standard emerged from Sun Microsystems, the Java Message
10   Service (JMS). At a programming interface level, this standard describes how a
messaging middleware is accessed from a piece of Java code. The two main
abstractions are "topics" (publish / subscribe messaging) and "queues" (point-to-point
messaging). While the standard describes the interface to the messaging
middleware, the implementation of the middleware is not specified. Also, integration
15   of non-Java programming languages is not specified, and integration of non-
programmable messaging devices (such as phones or pagers) are not specified.

Existing messaging middleware allows one to access the middleware over a fixed,
small number of transport protocols. These are usually TCP or SSL. Supporting a
new protocol requires the vendor of the middleware to implement it and to integrate it
20   into the middleware. Non-Java devices may or may not be supported, but again,
extending the middleware for as yet unsupported devices requires the vendor of the
middleware to enable it. Non-programmable devices are unsupported in current
messaging middleware products.

This leads to:

o Performance impacts, as TCP or SSL are unicast transport protocols, while the publish/subscribe pattern is a multicast abstraction.

o Limited applicability, as devices using a new kind of transport protocol are not supported. This is especially true for wireless small devices, such as PDAs or mobile phones, which today cannot be part of a uniform messaging infrastructure.

o Limited applicability, as non-programmable devices (such as a mobile phones with a built-in messaging function, e.g. Short Message Service (SMS)) cannot participate in a uniform messaging infrastructure.

o A limited choice of message delivery guarantees, resulting in potentially too strong guarantees (which are too expensive) or too weak guarantees (too many messages are lost undetected). This is inadequate especially for wireless networks.

o No support of asymmetrical networks, such as a cheap, best-effort bulk downlink for the actual delivery of messages, and a more expensive, reliable uplink for control data to implement the desired quality of service, i.e. message delivery guarantee.

Object of the invention:

A first object of the invention is to provide a messaging system for the delivery of data in the form of portable message formats between message clients using transport protocol adapters and any kind of network. Another object of the invention is to provide a method for messaging simple logical communication topics or message queues that are independent of transport protocols, delivery guarantees, message format, and message content limitations. Another object of the invention is to provide a computer program product directly loadable into the memory of a computer usable for running a messaging system.

Summary of Invention:

The messaging system outlined in this disclosure is a major technological advancement enabling users to deliver messages over any transport protocol, using an optimized message delivery guarantee, and to any kind of device.

5   A messaging system is disclosed for delivering data in the form of portable message formats from a producer of any kind, over any transport protocol, using any delivery guarantee, to one or more recipients of any kind. The method for running said message system includes a message broker with a system architecture of at least one pluggable protocol adapter. Said system architecture may comprise also at least

10   one pluggable message format adapter and at least one pluggable message content adapter, thus enabling to use a simple unified topic or queue abstraction between the involved communication parties.

Specifically, the method includes protocol adapters, message format adapters, and message content adapters to wireless networks and devices, as well as message

15   adapters to convert the portable message between the different formats used in different computer programming languages. Data may be delivered using asymmetric networks where the forward and return channels may be realized using two simplex channels, and networks with only a unidirectional channel.

The invention comprises also a computer program product directly loadable into the

20   memory of a computer usable for running a messaging system.

Detailed description Of The Drawing:

Fig. 1 provides a block diagram of a preferred embodiment of the present invention. It consists of:

- o   A message server **1a**.

5 o   A Java message client **2a**, connected via an IP multicast connection **1**.

- o   A Thin Java message client **2b**, connected over an asymmetric wireless transport protocol **2**.

- o   A Thin Java message client, **2c**, connected over a wireless transport protocol **3**.

- o   A Non-Java message client **2d**, connected over a Network **6** link to the message
10   format adapter **3a**.

- o   A non-programmable message client **2e**, connected over a telecommunications network (e.g. GSM) **7** to the message content adapter **4a**.

- o   A message format adapter **3a**, connected to the message server via an HTTP connection **4**.

15 o   A message content adapter **4a**, connected to the message server via a TCP connection **5**.

The block diagram is but one example of a messaging infrastructure deployment. Any number of message servers, message clients, message format adapters and message content adapters can be present in a specific installation.

20   The message server **1a** maintains client connections, administers client subscriptions to topics and queues, receives and forwards messages, and stores persistent

messages in its database. These activities are standard activities of a message server.

The message server **1a** comprising at least one pluggable transport protocol adapter. **Fig. 1** shows an example of six specific transport protocol adapters (UDP, SSL, HTTP, TCP, WAP, DAB/GSM Data). A placeholder is present (labeled "Other") for any number of additional protocol adapters.

On startup, the message server **1a** reads its configuration data, and initializes all configured protocol adapters. At runtime, additional protocol adapters can be started, or running protocol adapters can be stopped, without interrupting the message server service (however, if a specific protocol adapter is stopped, service over this adapter is no longer available).

At least one message client **2a-2e** connects to the message server **1a** using its configured protocol adapter. If a matching protocol adapter is running on the server, the connection is successful. Further communication between message client and message server is according to the familiar publish/subscribe or point-to-point pattern of JMS.

JMS topics or JMS queues are named and administered independently of the protocol adapters involved. If a client connects to the server using the "TCP" protocol adapter, it can communicate with a client that connected using the "UDP" protocol adapter, if both clients use the same JMS topic or queue.

The protocol adapters encapsulate at least one logic needed to:

o   Interface with a transport protocol, such as HTTP, TCP or GSM Data.

o   Specify and guarantee a quality of service for the message delivery.

Certain transport protocols operate in a "best effort" delivery mode. Thus, simply adapting to a specific protocol is not always enough (unless "best effort" is the

desired message delivery guarantee). Thus, protocol adapters consist of both the transport protocol mechanism, and a quality of service mechanism to improve the basic network delivery guarantee.

The Java message clients **2a-2c** implements the JMS API from Sun Microsystems. It
5   cooperates with the message server to achieve full JMS functionality.

Specific to the present invention is the pluggable transport protocol adapter architecture. A message client **2a-2e** can use any transport protocol adapter to communicate with the message server **1a**. The necessary code to implement a specific transport protocol adapter can be acquired at runtime, for example using a
10   Java classloader mechanisms, using a lookup service such as JINI, or using a directory service via JNDI.

The thin Java message client **2b,2c** is a compact version of a Java message client. It is able to operate without any network support from the Java environment. Specifically, the java.net, java.io and java.rmi libraries are not required. This is
15   possible because of the pluggable transport protocol adapter architecture. The protocol adapter to access e.g. an infrared port on a mobile device directly interfaces with the infrared hardware. Thus, it does not require any TCP/IP emulation on top of the infrared hardware to be present.

The Non-Java message client **2d** connects for example via a TCP link to the
20   message format adapter. It consists of a JMS-like programming library, that offers to non-Java devices JMS functionality, as well as full integration with Java message clients.

The non-programmable message client **2e** is any device that has its own method to interchange messages with other devices. Examples include:

25   ⊚   GSM Phones, using the Short Message Service (SMS).

     ⊚   Alphanumeric pagers.

o WAP enabled phones.

A non-programmable message client **2e** needs for example a specific message content adapter to integrate into the messaging system, i.e. to exchange messages with other client types.

5   The JMS message format is specific to Java and JMS. Non-Java programmable clients need a translation engine to convert the Java message into a message they can deal with. This translation engine is the message format adapter **3a**. According to **Fig. 1**, the message format adapter **3a** is a generic proxy. On one side, it connects to the message server **1a** using any pluggable transport protocol adapter. On the other

10   side, it communicates with the Non-Java message client **2d** using a command protocol over a communication such as TCP/IP. The command protocol supports the entire JMS command set, such as "publish" or "subscribe". JMS object messages are introspected using the Java introspection mechanism. The result of the introspection is translated into a byte stream and exchanged with the Non-Java message client **2d**

15   over the TCP/IP link.

Non-programmable clients usually have restrictions regarding the size and format of the messages they support. For example, the GSM SMS specification allows only text, with a maximum length of 160 characters. Additionally, the non-programmable clients don't follow JMS, or the publish/subscribe or point-to-point programming

20   pattern. Thus, other means for e.g. topic registration must be used.

The message content adapter **4a** is described generically, but must be specifically adapted to each non-programmable message client. There is e.g. a SMS message content adapter, a WAP message content adapter or a pager message content adapter.

25   According to **Fig. 1**, the message content adapter **4a** is a proxy that on one side connects to the message server **1a** using any pluggable transport protocol adapter. On the other side, it communicates either directly with the non-programmable

message clients, or with existing telecommunications equipment (such as an SMS gateway) that then communicates with the non-programmable message clients.

Topic subscription is handled with a specific command protocol. In some cases, this command protocol must be known to the user (e.g. with SMS a message
5   "subscribe:/news/sports/" needs to be sent to a specific service number). In other cases, the subscription command can be hidden from the user (e.g. with WAP, the actual subscription command can be hidden below a descriptive link).

Message adaptation is done with knowledge of the properties of the non-programmable message client. E.g. a GSM phone using SMS needs to receive a text
10   string of at most 160 characters. An incoming Java message is thus introspected, and text content is extracted. If the text content is more than 160 messages, it is truncated. In the other direction, a published SMS message is converted into a JMS text message. Likewise scenarios exist for other non-programmable message clients.

As the messaging system comprises a computer, a computer program product
15   comprising a software code for performing the steps of the invention can be directly loaded into the memory of said computer of the messaging system.

One example of such a messaging system is applicant's SoftWired iBus//MessageServer computer program product. This computer program product is being delivered in the form of an installable software packet. This software packet is
20   installed on a hardware server such as a common Intel-based PC or a Sun Sparc based Unix computer. This computer program product is directly loadable into the memory of such a computer. After installation, a message server **1a** can be started. As part of the message server, at least one pluggable transport protocol adapters is being delivered. Transport protocol adapters are in the form of a set of software
25   components, i.e. Java classes. One example is the protocol adapter for the "reliable IP multicast" transport, which consists of the Java classes DISPATCH, FRAG, FIFO, NAK, REACH and IPMCAST. By configuring/implementing the above classes in the proper order in the message server configuration file, the "reliable IP multicast" transport adapter is initialized and started/plugged at runtime by the message server.

In summary, any number of protocol adapters can be activated in the message server by specifying them in the server configuration file and by delivering and installing the necessary Java classes to implement the protocol adapter.

Java Message Clients or Thin Java Message Clients implement the necessary logic
5   in the form of a of a library, i.e. a Java Archive (JAR). This Java Archive is installed and started on a client computer. Protocol adapters are implemented and configured the same way as on the server, i.e. configured in a configuration file and delivered in the form of Java classes. The Java classes can be part of the Java Archive or can be downloaded at runtime from a web server.

10 A Message Format Adapter is in essence a special form of a Java Message Client. Thus, it consists of the same components as a Java Message Client and is configured the same way. In addition, it contains programming logic to reformat and route messages from and to the non-Java Message Clients.

A Message Content Adapter is in essence also a special form of a Java Message
15 Client. Thus, it consists of the same components as a Java Message Client and is configured the same way. In addition, it contains programming logic to analyze, reformat and route messages from and to the non-programmable Message Clients.

Glossary of terms used

TCP: Transmission Control Protocol
UDP: User Datagram Protocol
IP: Internet Protocol
5   HTTP: Hypertext Transfer Protocol
WAP: Wireless Application Protocol
SSL: Secure Socket Layer
JMS: Java Message Service (http://java.sun.com/products/jms/)
PDA: Personal Digital Assistant
10   SMS: Short Messaging Service
GSM: Global System for Mobile Telecommunication
DAB: Digital Audio Broadcast